Algoritmos y Estructuras de Datos

Segundo Parcial 2020

Scheduler de Tareas Repetitivas

Realizar un programa que ejecute en forma cronológicamente ordenada las actividades planificadas por distintos tipos de “relojes”.

La ejecución es simbólica, lo que importa es el lanzamiento de la actividad en el momento que cada una de ellas debe ser lanzada.

Para esto tomará como entrada desde un archivo la configuración de las actividades planificadas por cada reloj.

Formato de entrada de relojes: el archivo se compondrá de líneas en la cual cada línea representa un reloj. Cada línea tiene la información separado por espacios, en el siguiente órden

- Número de reloj : número entero positivo
- Nombre del reloj : string
- Tipo de reloj: un *string* con dos posibles valores (aleatorio/periódico)
- Tiempo de repetición: es un número entero en segundos, el cuál representa el tiempo de repetición de los eventos generados por ese Reloj. En el caso de los eventos periódicos representa el tiempo de repetición. Si es aleatório, representa el máximo tiempo que puede tardar en repetirse un evento aleatorio (entre 1 y n, siendo n el máximo).

Ej:

1 Reloj_1 periodico 5

2 Reloj_2 aleatorio 3

3 Reloj _3 periodico 17

4 Reloj _4 aleatorio 15

Debe tener por lo menos 10 relojes, 5 aleatorios y 5 periódicos. Es recomendable que sean más de 10 (se verá más adelante).

El contenido del archivo es leído y por cada linea se inicializa un objeto “Reloj” que posee las siguientes cualidades:

- Debe poder generar eventos: De acuerdo a su configuración el reloj producirá objetos del tipo “Evento” que sucederán en un tiempo determinado acorde a su lapso de repetición.
- Si el reloj es de tipo periódico, generará eventos de intervalos constantes. Si es aleatorio, generará eventos a intervalos aleatorios.

A su vez, debe existir una clase “Planificador” o *Scheduler* que tenga estos métodos:

- agregarReloj : Agrega un reloj al planificador. Los relojes generaran eventos.
- getProximoEvento: Devuelve el evento más próximo a ser ejecutado.
- run: Es un ciclo que cumple la siguiente lógica.
    - Obtiene el siguiente evento a ejecutarse.
    - Duerme (sleep) la cantidad de segundos que el evento indica que quedan para ejecución.
    - Ejecuta el evento (explicado más adelante)
    - Lleva la cuenta de cuanto eventos son lanzados. Cada 500 eventos lanzados (y tambien al comienzo del programa), le pide a cada reloj que genere 50 nuevos eventos, los cuales son nuevamente planificados. Por esto es conveniente tener más de 10 relojes, para que el planificador no quede vacio.

Deben definirse 3 clases Planfiicadores de eventos que guarden los eventos y los mantengan ordenados por su momento de lanzamiento. El primero en una lista enlazada, el segundo en un montículo (heap) implementado sobre un arreglo (esta estructura es conocidad como cola de prioridad o priority queue) y el tercero con un montículo binomial (binomial heap).

El Planificador con la lista los mantendrá ordenados por el el algoritmo de Quicksort, los otros dos métodos tienen de por sí un órdenamiento propio de este tipo de schedulers. Por cada scheduler debe contar la cantidad de comparaciones que hace y poder presentar al final del programa la eficiencia de cada uno de estos.

A su vez, por cada uno de estos eventos generados:

- Debe poder “ejecutarse”: en este caso significa imprimir en un archivo la hora (con precisión del segundo) seguido del nombre del evento. El nombre de Evento se forma a partir del nombre del reloj y el número del evento.
- Se debe poder calcular el tiempo restante al evento. Se guarda la hora en cada ejecución y se debe poder calcular en base al tiempo actual cuánto falta para el siguiente evento.

Ej de salida de las “ejecuciones” en el archivo:

```
5-20-2020 19:52:03 Reloj_2 Evento 1
5-20-2020 19:52:05 Reloj_1 Evento 1
5-20-2020 19:52:06 Reloj_2 Evento 2
5-20-2020 19:52:09 Reloj_2 Evento 3
5-20-2020 19:52:10 Reloj_4 Evento 1
5-20-2020 19:52:10 Reloj_1 Evento 2
5-20-2020 19:52:12 Reloj_2 Evento 4
5-20-2020 19:52:15 Reloj_1 Evento 3
5-20-2020 19:52:15 Reloj_2 Evento 5
5-20-2020 19:52:16 Reloj_4 Evento 2
5-20-2020 19:52:17 Reloj_3 Evento 1
```

El funcionamiento del **main** del programa es básicamente leer los clocks del archivo, ejecutar el run() del Planificador hasta que su ejecución sea interrumpida (Ctrl + C). Cuando acaba la ejecución debe informar la cantidad de comparaciones de cada scheduler.

Recomendamos estos sitios para estudiar la estructura de binomial heap:

https://algorithmtutor.com/Data-Structures/Tree/Binomial-Heaps/

http://webdiis.unizar.es/asignaturas/TAP/material/3.3.binomiales.pdf

https://www.youtube.com/watch?v=7UQd9SYUoNk

Y el siguiente video como repaso del montículo

https://www.youtube.com/watch?v=AD_J4ZUheik